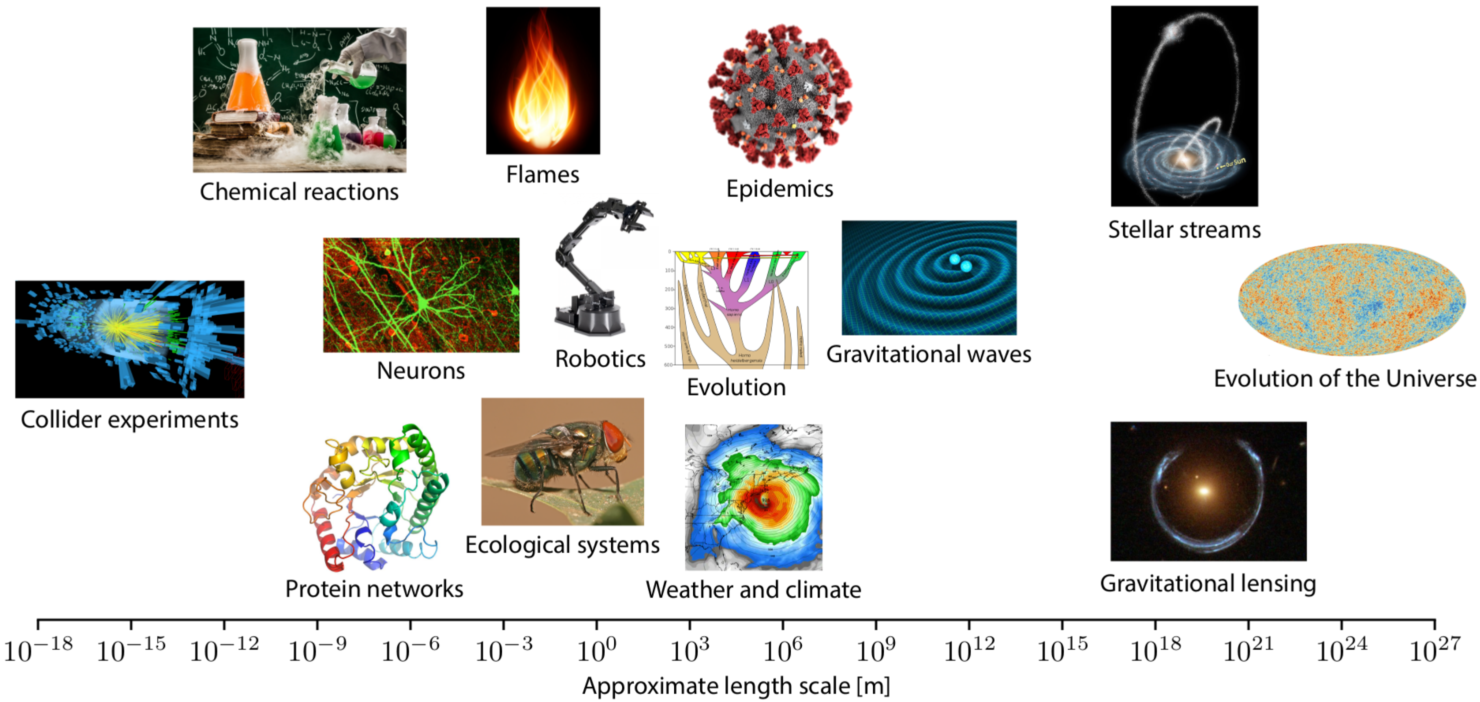
Scaling AI for Probabilistic Programming in Scientific Simulators

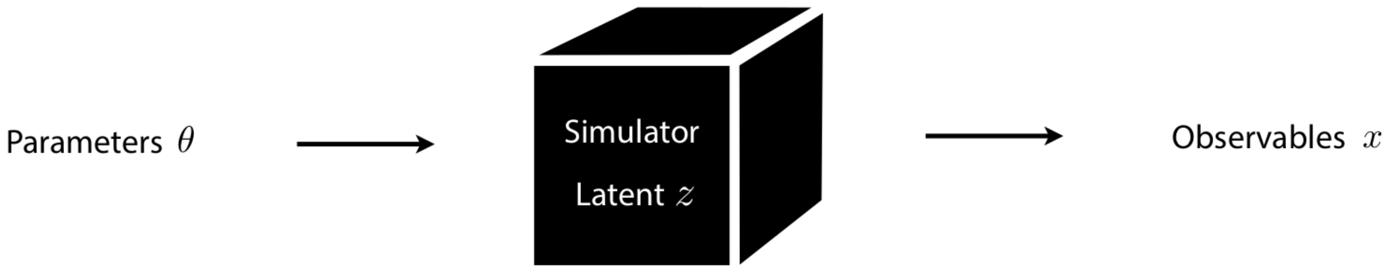
EuroHPC's LUMI kick-off, Belgium
January 14, 2021

Gilles Louppe
g.louppe@uliege.be



Credits: Johann Brehmer.

Simulation-based inference



Prediction:

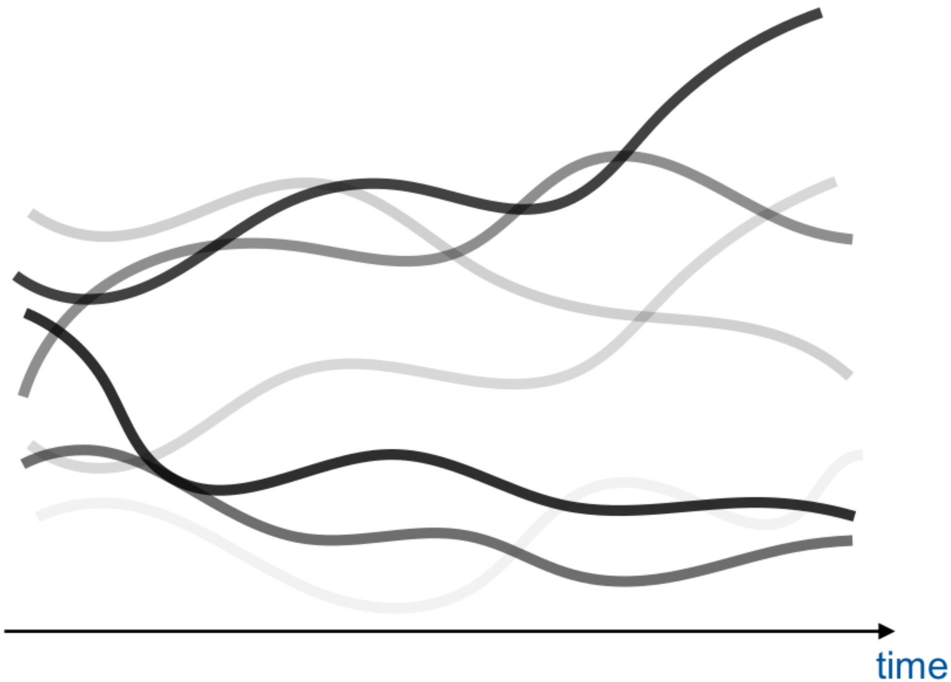
- Well-motivated mechanistic, causal model
- Simulator can generate samples $x \sim p(x|\theta)$

Inference:

- Interactions between low-level components lead to challenging inverse problems
- Likelihood $p(x|\theta) = \int \mathrm{d}z \; p(x, z|\theta)$ is intractable

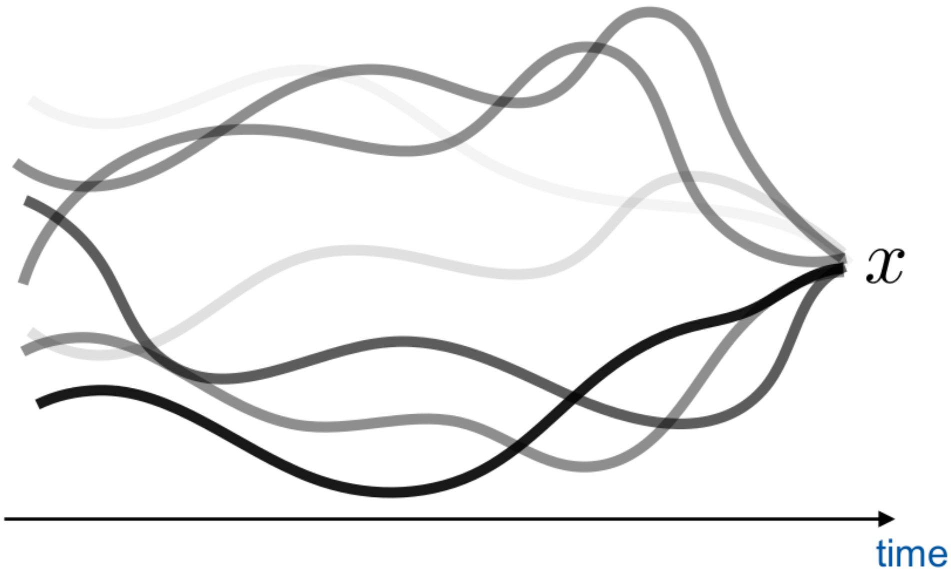
—
Credits: Johann Brehmer.

Bayesian inference



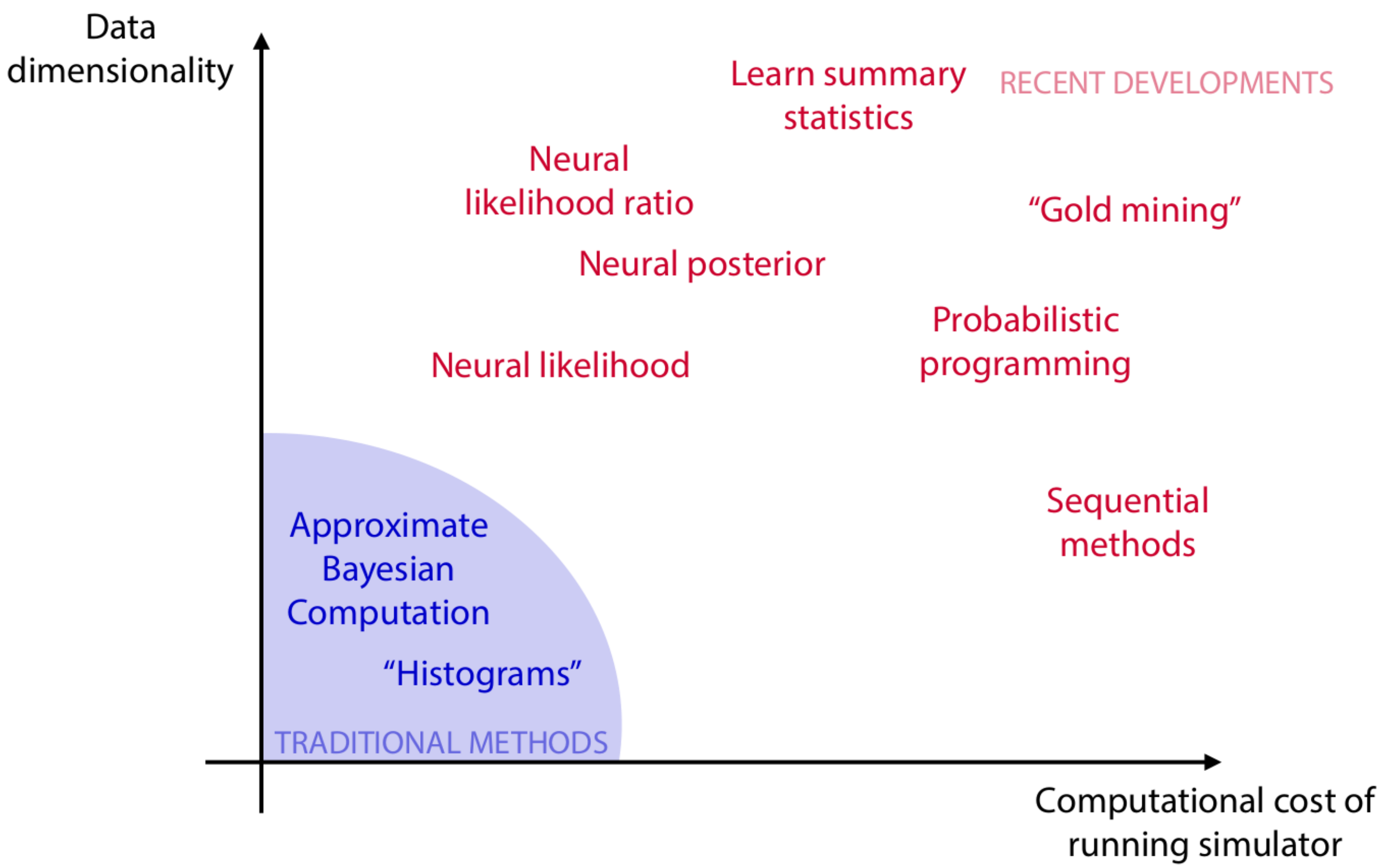
Unconditioned probabilistic program.

$$\theta, z, x \sim p(\theta, z, x)$$



Conditioned probabilistic program.

$$\theta, z \sim p(\theta, z | x)$$



Credits: Johann Brehmer.

Probabilistic programming

Probabilistic programming is a machine learning framework allowing us to

- write programs that define probabilistic models;
- run automated Bayesian inference of parameters conditioned on observed outputs (data).

Pyro Edward Stan

> Probabilistic programming normally requires one to implement a probabilistic model from scratch, in the chosen language/system.

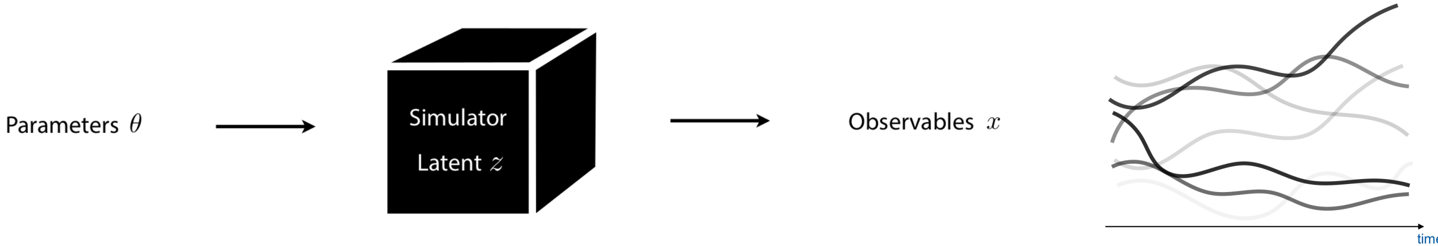
Key idea

Many HPC simulators are stochastic and they define probabilistic models by sampling random numbers. **Scientific simulators are probabilistic programs!**

We "just" need an infrastructure to execute them as such.

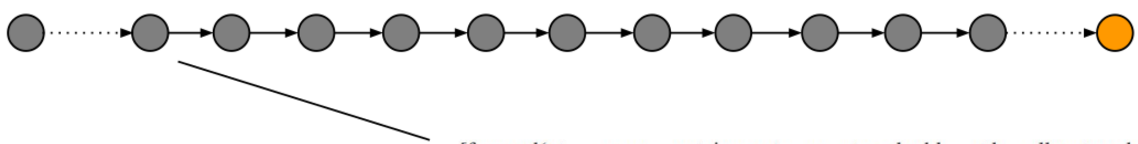
PyProb

A new probabilistic programming system
for simulators and HPC, based on PyTorch.

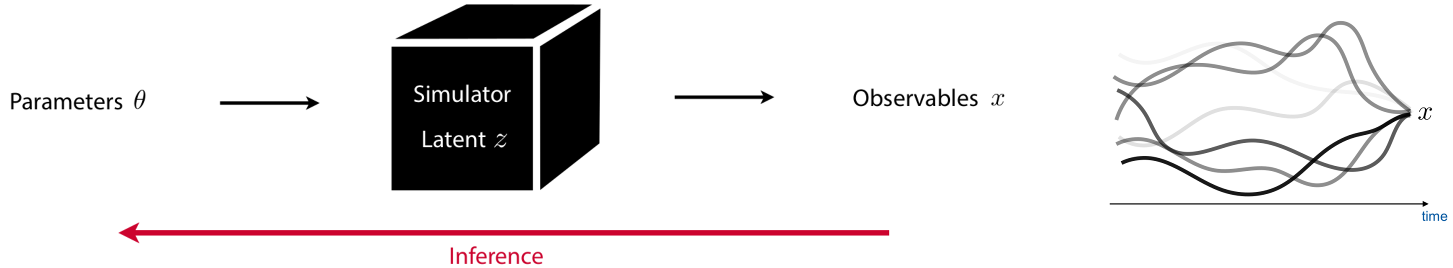


Forward execution

- Run forward and catch all random choices ("hijack" all calls to RNGs).
- Record an execution trace: a record of all parameters, random choices, outputs

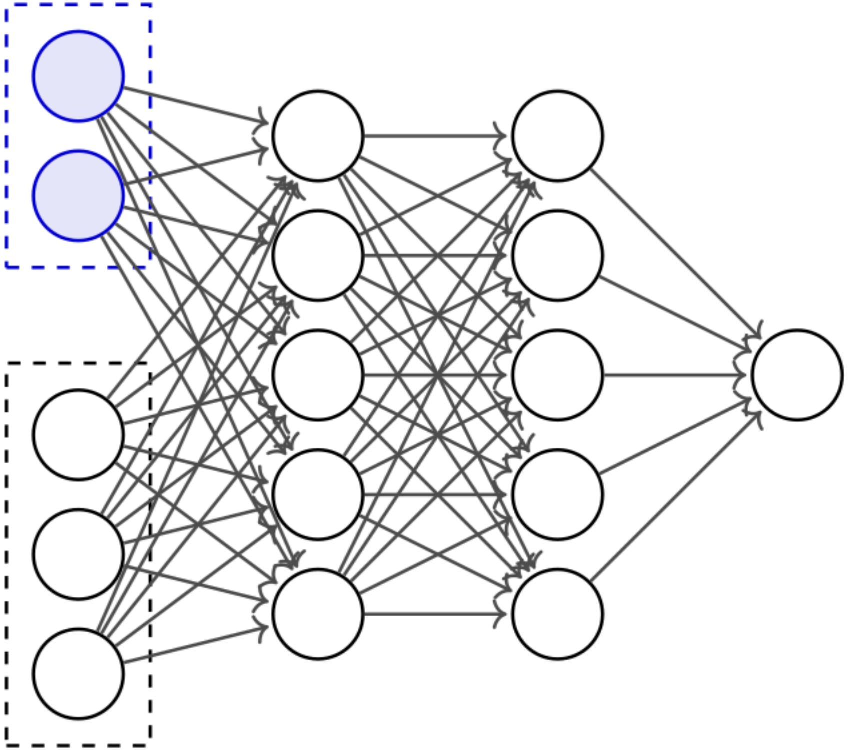


PPX

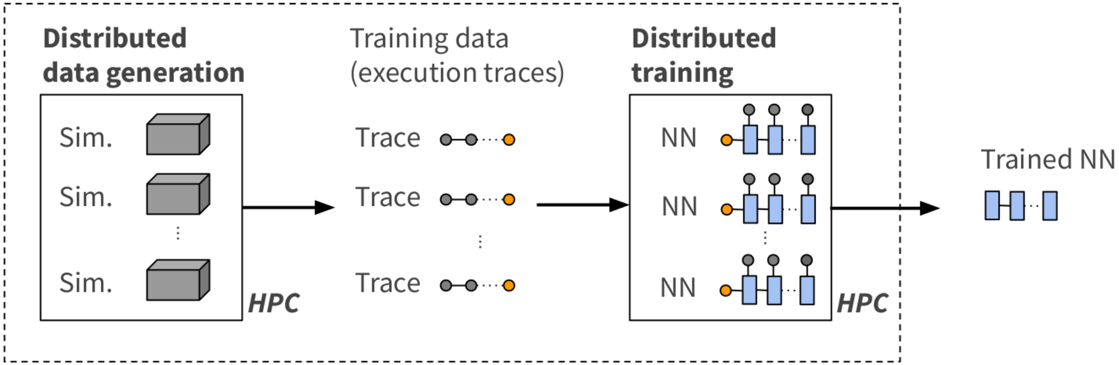


Inference

- Approximate the distribution of parameters that can produce (explain) observed data, using inference engines like MCMC.
- This is hard and computationally costly.
    - Need to run simulator up to millions of times
    - Simulator execution and MCMC inference are sequential
    - MCMC has "burn-in" and autocorrelation.



Good news: We can amortize the cost of inference using deep learning.

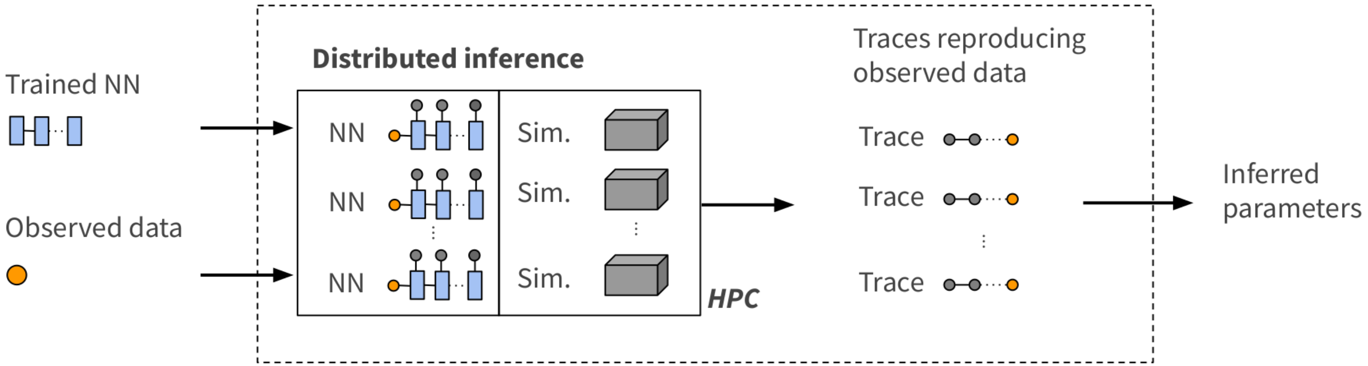


Training (recording simulator behavior)

- Deep recurrent neural network learns all random choices in simulator.
- Dynamic NN: grows with simulator complexity
  - Layers get created as we learn more of the simulator.
  - 100s of millions of parameters
- Costly, but amortized: we need to train only once per given model

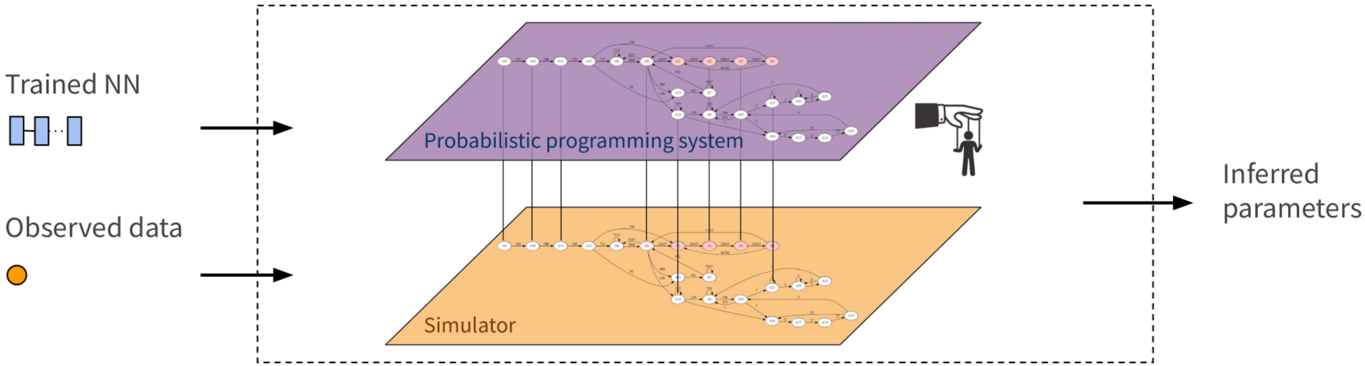
Inference (controlling simulator behavior)

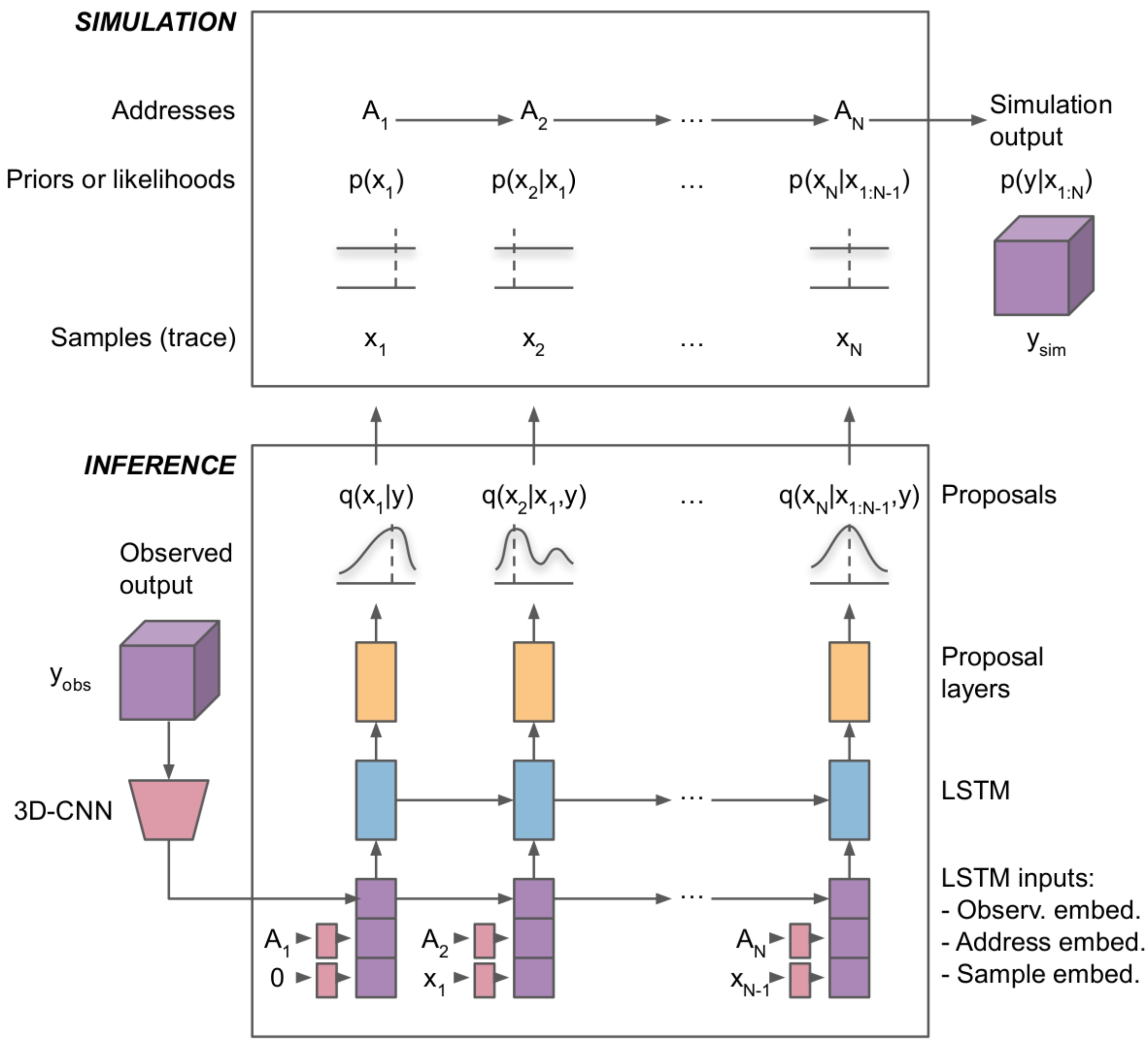
- Trained deep NN makes intelligent choices given data observation
- Embarassingly parallel distributed inference
- No "burn-in period". No autocorrelation.



Inference (controlling simulator behavior)

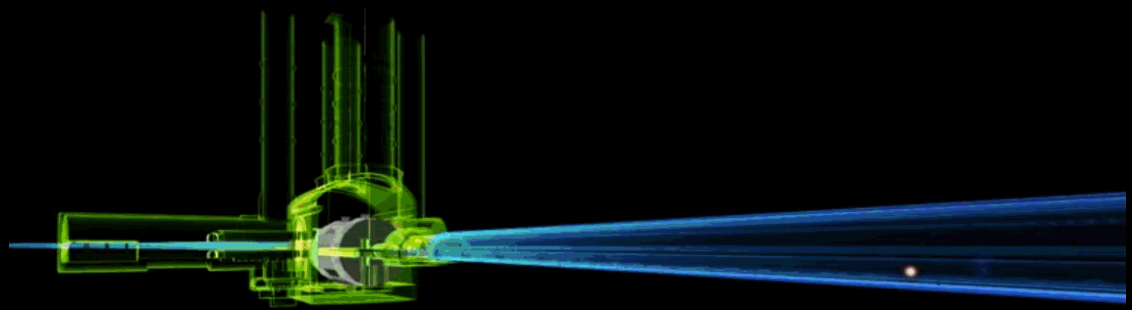
- Trained deep NN makes intelligent choices given data observation
- Embarassingly parallel distributed inference
- No "burn-in period". No autocorrelation.




SIMULATION
Addresses
A_1
A_2
...
A_N
Simulation output
Priors or likelihoods
$p(x_1)$
$p(x_2|x_1)$
$p(x_N|x_{1:N-1})$
$p(y|x_{1:N})$
Samples (trace)
x_1
x_2
x_N
y_{sim}
INFERENCE
$q(x_1|y)$
$q(x_2|x_1,y)$
$q(x_N|x_{1:N-1},y)$
Proposals
Observed output
y_{obs}
Proposal layers
LSTM
3D-CNN
LSTM inputs:
- Observ. embed.
- Address embed.
- Sample embed.
0
x_1
x_{N-1}

Use case: LHC

Inverting the Large Hadron Collider

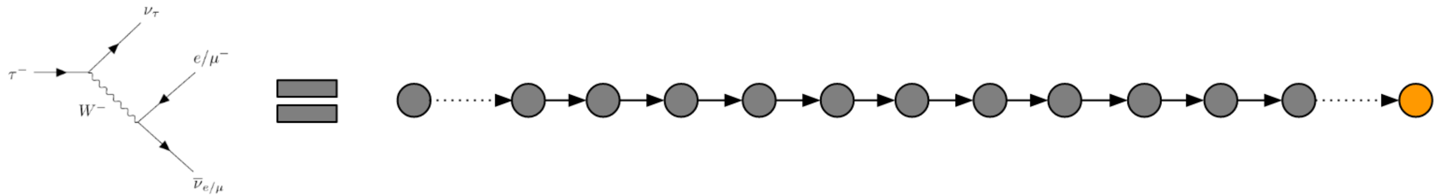


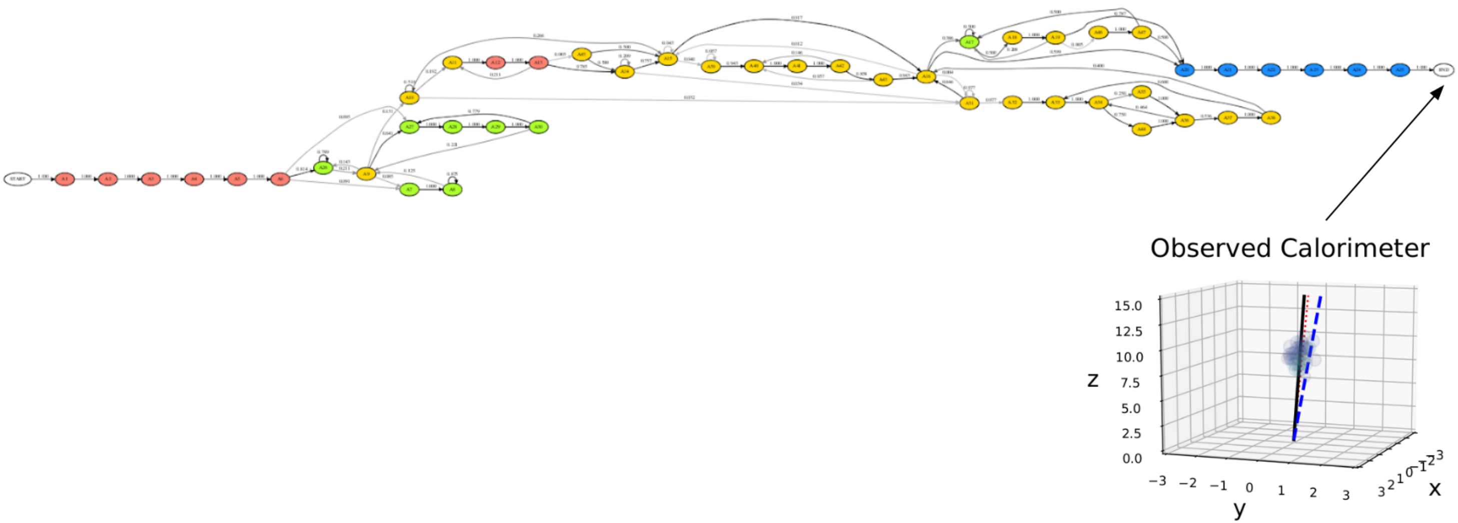
Large Hadron Collider

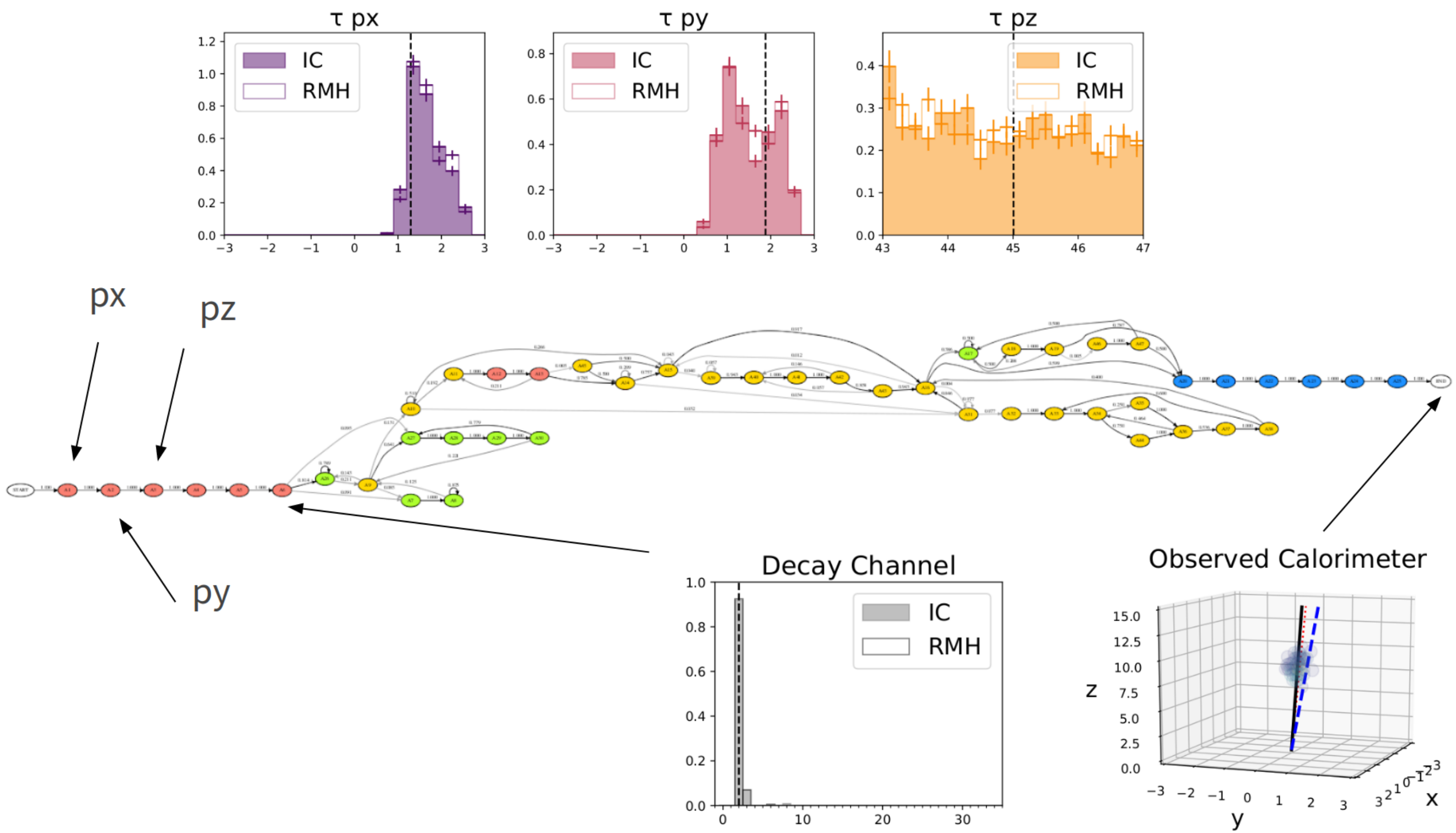
- Seek to uncover secrets of the universe (new particles).
- Today, physicists compare observed data to detailed simulations, using billions of CPU hours for scans of simulation parameters (inefficient, labor-intensive, sometimes ad-hoc).
- PyProb replaces this with automated, efficient inference; grounded in a statistical framework.

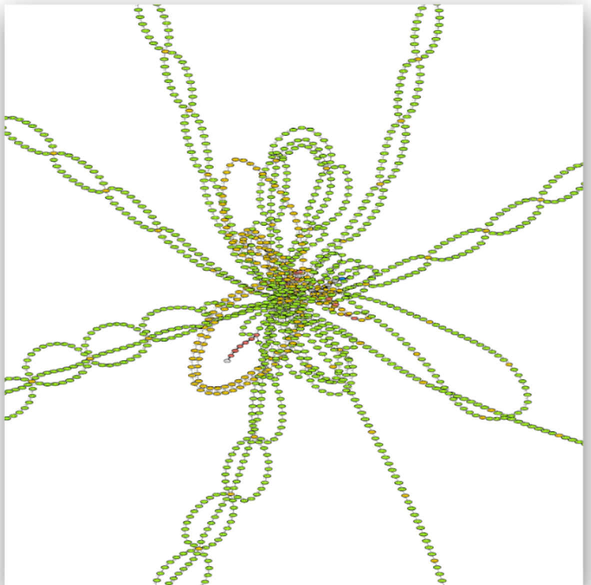
Physics expressed in simulator code

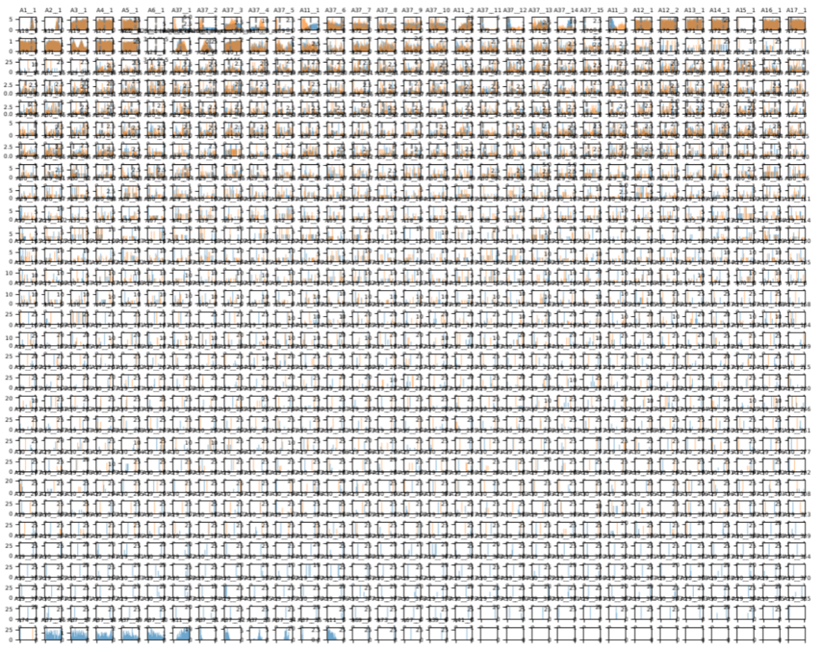
- We base on our proof-of-principle on existing Sherpa simulation (1M lines of C++ code).
- Execution traces represent particle physics collisions and decays.
- PyProb will enable interpretability by relating a detector observation to possible traces that can be related back to the physics.




Observed Calorimeter
15.0
12.5
10.0
7.5
5.0
2.5
0.0
z
−3 −2 −1 0 1 2 3
y
3 2 1 0 −1 2 3
x

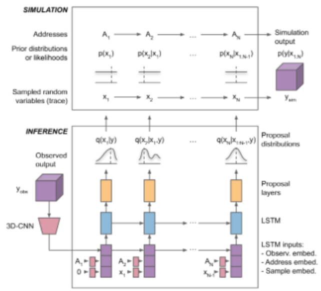

τ px
IC
RMH
τ py
IC
RMH
τ pz
IC
RMH
px
pz
py
Decay Channel
IC
RMH
Observed Calorimeter
z
y
x

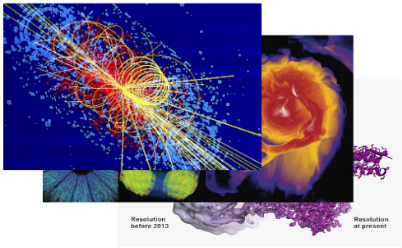


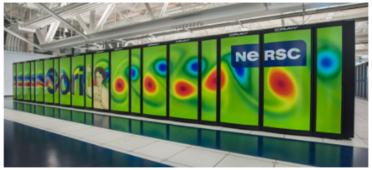


PyProb gives access to all latent variables:
allows answering *any* model-based question.

Reaching supercomputing scale







Probabilistic programming

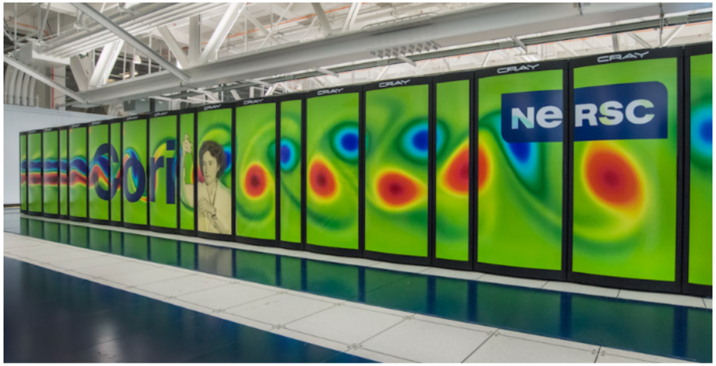
Simulation

Supercomputing

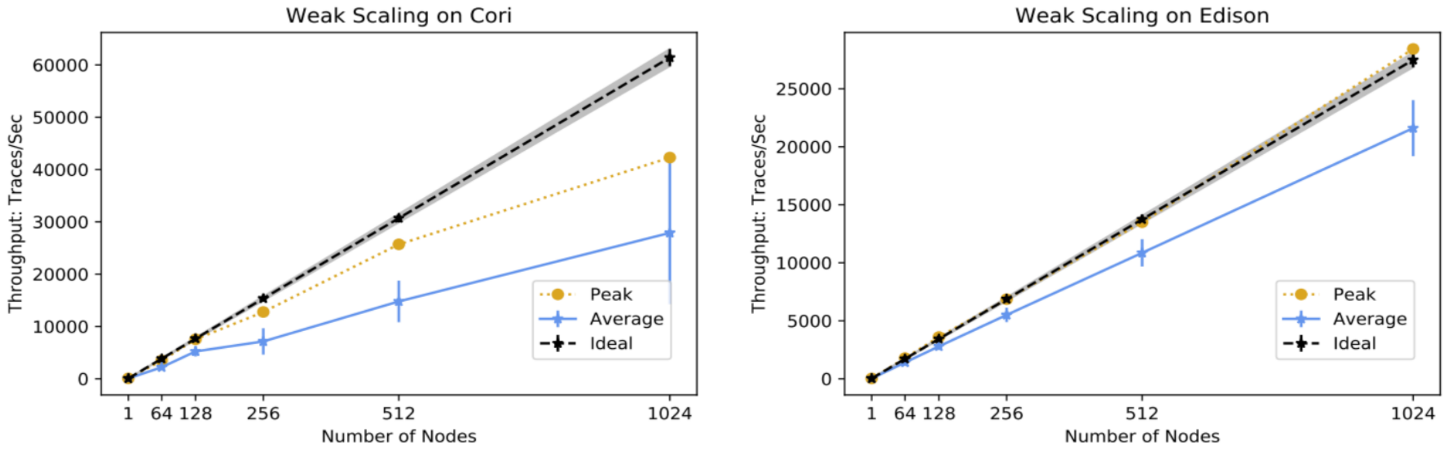
Need for HPC resources and considerable optimization,
for both simulation and NN training.

Platforms and experimental setup

- NERSC Cori Cray XC40
- NERSC Edison Cray XC30
- Intel Diamond Cluster



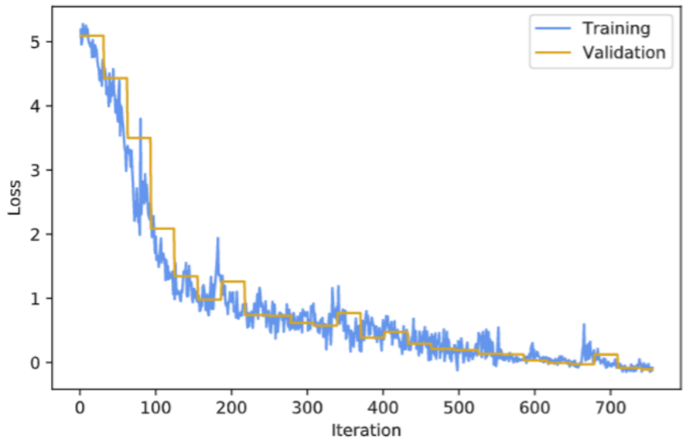
Scaling

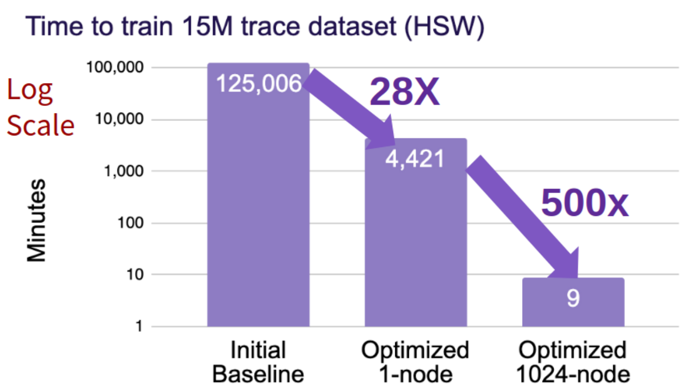


Weak Scaling on Cori
Throughput: Traces/Sec
0
10000
20000
30000
40000
50000
60000
1
64
128
256
512
1024
Number of Nodes
Peak
Average
Ideal
Weak Scaling on Edison
Throughput: Traces/Sec
0
5000
10000
15000
20000
25000
1
64
128
256
512
1024
Number of Nodes
Peak
Average
Ideal

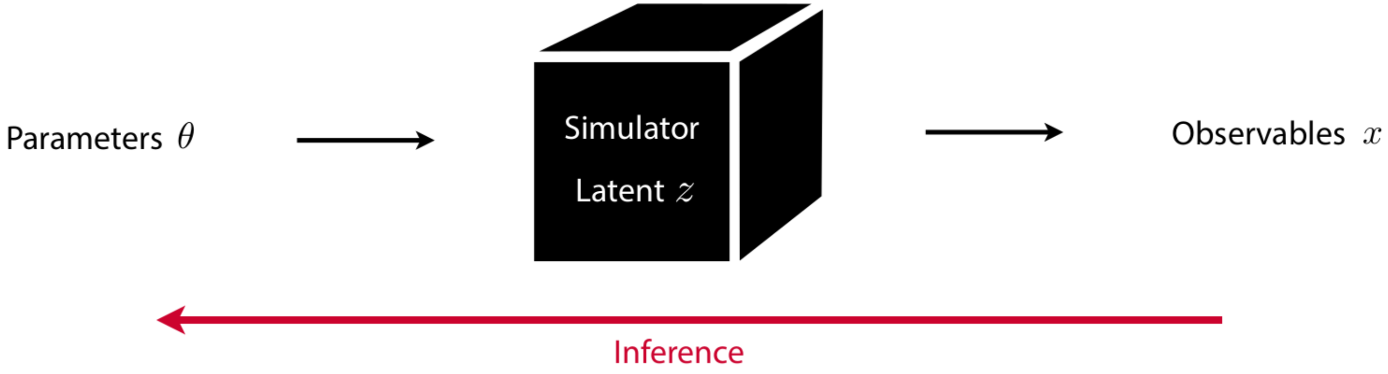
Large-scale training

- Dataset of 15M Sherpa execution traces (1.7 TB)
- Fully synchronous data parallel training on 1024 nodes (32768 cores) using PyTorch-MPI
- Global mini-batch size of 128000
- Overall 14000x speedup
    - Months of training in minutes
    - Ability to retrain model quickly is transformative for research

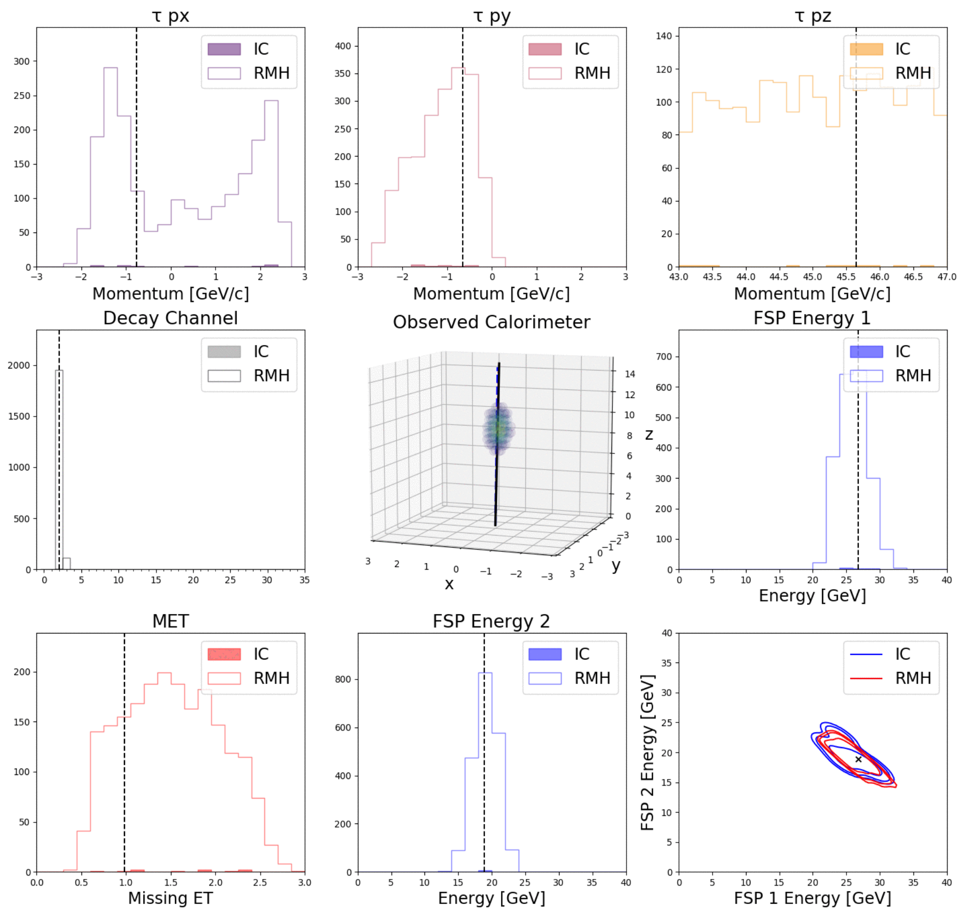




Reminder: We are doing all this to performance inference.



Science results

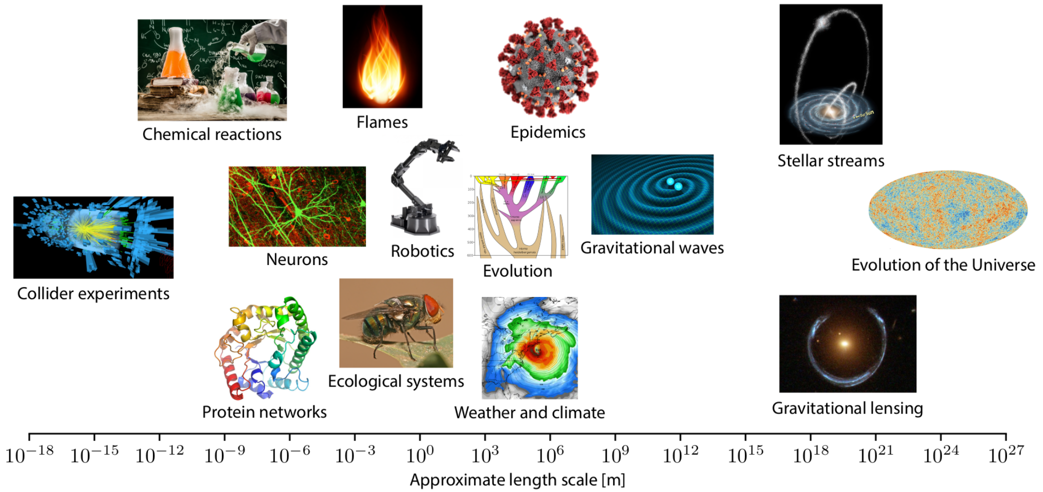


First tractable Bayesian inference for LHC physics
(Full posterior and interpretability)

Summary

- PyProb is a probabilistic software framework to execute and control exisiting HPC simulator code bases.
- Synchronous data parallel training of a NN is made possible thanks to HPC.
- AI-powered probabilistic programming is for the first time practical for large-scale, real-word science models.

This is just the beginning...



A team effort

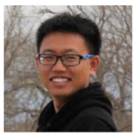

Atılım Güneş Baydin


Lei Shao


Wahid Bhimji


Lukas Heinrich


Lawrence Meadows

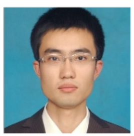

Jialin Liu

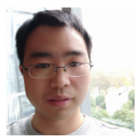

Andreas Munk


Saeid Naderiparizi

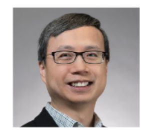

Bradley Gram-Hansen


Gilles Louppe


Mingfei Ma


Xiaohui Zhao


Phil Torr


Victor Lee


Kyle Cranmer


Prabhat


Frank Wood

UNIVERSITY OF OXFORD CERN NYU UBC BERKELEY LAB NERSC intel

References

- Baydin, A. G., Shao, L., Bhimji, W., Heinrich, L., Meadows, L., Liu, J., ... & Ma, M. (2019). Etalumis: Bringing Probabilistic Programming to Scientific Simulators at Scale. arXiv preprint arXiv:1907.03382.
- Baydin, A. G., Heinrich, L., Bhimji, W., Gram-Hansen, B., Louppe, G., Shao, L., ... & Wood, F. (2018). Efficient Probabilistic Inference in the Quest for Physics Beyond the Standard Model. arXiv preprint arXiv:1807.07706.
- Casado, M. L., Baydin, A. G., Rubio, D. M., Le, T. A., Wood, F., Heinrich, L., ... & Bhimji, W. (2017). Improvements to Inference Compilation for Probabilistic Programming in Large-Scale Scientific Simulators. arXiv preprint arXiv:1712.07901.